CS534 - Machine Learning

Implementation Assignment 3

Group 50
Sebastian Mueller - 933962290
Derek Helms - 934451909
Vy Bui - 934370552

Instructor: Dr. Xiaoli Fern

The School of Electrical Engineering and Computer Science
Oregon State University

Nov 7, 2022

Part 0

Question 0a:
Please report the ten most frequent words (frequency is measure by the total counts of each word are maximal) for the positive tweet and negative tweet respectively. Do you find these words to be informative of the classes?

Response:

Word	**Count**
the	746
to	716
you	716
for	504
thanks	462
jetblue	455
southwestair	452
united	414
thank	355
and	330

Table 1: Ten most frequent words by count for positive tweets

Word	**Count**
to	4741
the	3236
flight	2313
united	2254
on	2202
and	2191
you	2130
for	2104
my	1926
usairways	1885

Table 2: Ten most frequent words by count for negative tweets

We found some words to be slightly informative, but in general most were not. For example, "thank", "thanks" and "you" might all be considered positive indicators since they showcase gratitude (e.g. "Thank you"). However words like "the", "to", and "and" are neutral words that would be used in positive and negative classes, so their high count really does not tell us much about the class. Additionally, the word "you" could fall in both classes. Positive when "thank you" and negative when "you" is used to accuse an airline of wrong doing (e.g. "You lost my bag!") These results are showcased in Table 1&2.

It is important to note though that these top ten counts may help discover patterns we were not initially aware of. For example, we initially thought "and" would tell us nothing about positive or negative classification, however we see it ranked higher in the negative class. Perhaps this is because people are more likely to list off everything negative about their experience opposed to everything positive about their experience, so "and" may be more informative of the negative class.

So in general, while these words are probably not robust metrics of classification, they do provide some information that is worthwhile to think about in the context of the problem.

Question 0b:
Please report the ten words with the highest total TF-IDF's for the positive tweet and negative tweet respectively. How do they compare to the list in 0(a)? Which one do you think are more informative and why?

Response:
We see similar words for both top ten positive and negative for TF-IDF and non TF-IDF, but they are differently ordered for TF-IDF (Table 3&4). Interestingly, the TF-IDF classification ranked "thanks", "thank" and "you" higher for the positive class than the non TF-IDF model, which makes sense to us. Those three words are going to be more informative of positive than "the", which the non TF-IDF model ranked first.

For the most part TF-IDF will be more informative. TF-IDF will give us a more accurate idea of "weight" for any word because TF-IDF offsets the count of any given word by the amount of samples that contains that word of interest. This will help normalize our weights, adjusting for the fact that certain words show up more often generally in usage.

Word	Count
you	110
thanks	91
thank	86
the	81
jetblue	81
united	73
to	73
southwestair	72
for	64
americanair	60

Table 3: Ten words with highest TF-IDF for positive tweets

Word	Count
to	359
the	295
flight	241
you	238
united	237
on	229
and	222
for	221
usairways	214
my	214

Table 4: Ten words with highest TF-IDF for negative tweets

Part 1

Question 1a:
1. What is the best validation performance you are able to achieve with linear SVM? What c value is used?

Response:
The best validation performance we were able to achieve (using accuracy as our performance metric) was 92.68%. The value of C used was .83125. This value was found by examining performance

of test and validation data for c values in range $10^{[-4,-3,\ldots,3,4]}$. This exploration revealed that the neighborhood of best c values was likely the range [0.1, 1]. Figure 2 highlights the results of that exploration where we isolated best c value for performance on validation accuracy.

Question 1b:
What trend do you theoretically expect to see for training and validation performance as we increase c? Plot the training and validation accuracy against different values of c. Does the trend you observe match your expectation?

Response:
Theoretically we expected to see both training and validation performance increase as we increased value of c. Eventually we expected to see a point where training performance continues to increase while validation performance began to decrease. This happened because as C values increase we are learning the specific margin best for our training data better; however, this margin may not generalize well (i.e. over fitting). The trend observed does match our expectation (Figure 1).

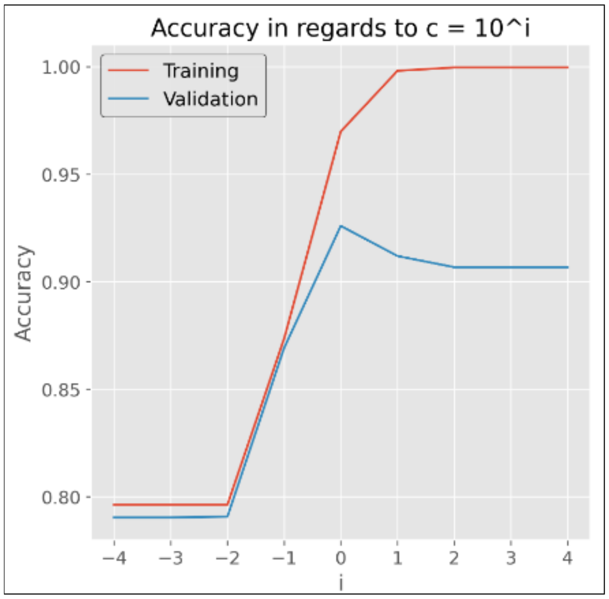


Figure 1: Accuracy of training and validation data for $c = 10^i$ for i in [.1,1]. Scatter plot on the right is scaled larger to better see results.

We were surprised at first to see both performance curves flatten out at a $i \approx 2$. We then realized that this happened because there is a "limit" of large C values where making the margin smaller will not do anything to improving training data performance. This will flatten our performance curve for training and in-turn also flatten our validation data. Interestingly, for our training data, since it flattened out at 100% accuracy, our training set used can have a separation boundary with no mis-classifications. This might not be the case for all training data used.

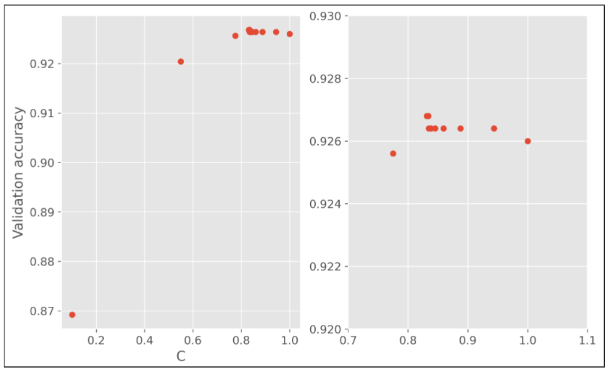


Figure 2: Accuracy of validation data for $c = 10^i$ for i in [.1,1].

Question 1c:
What relationship do we theoretically expect between c and the number of support vectors? Plot

the number of support vectors against the different values of c. Does the trend you see match your theoretical expectations?

Response:
We theoretically expect in general for there to be less support vectors for larger values of C. This is because as C increases we penalize the model more for mis-classifications, and thus our margin squeezes smaller and our boundary may shift to compensate. Essentially C determines how much slack our model allows. So as our margin gets smaller as we tolerate less slack, in general we will decrease the number of vectors that can actually fall inside or on the margin, and in-turn decrease the amount of support vectors we have. Our expectations matched our results (Figure 3). Changes in number of support vectors for fine tuning C experiment are provided in Figure 5 (Appendix)

** Important note. There are instances where increasing C could increase SV count. It would be data distribution dependent though. If the decision boundary shifts to decrease mis-classification, margins could hit a cluster of data not previously included.

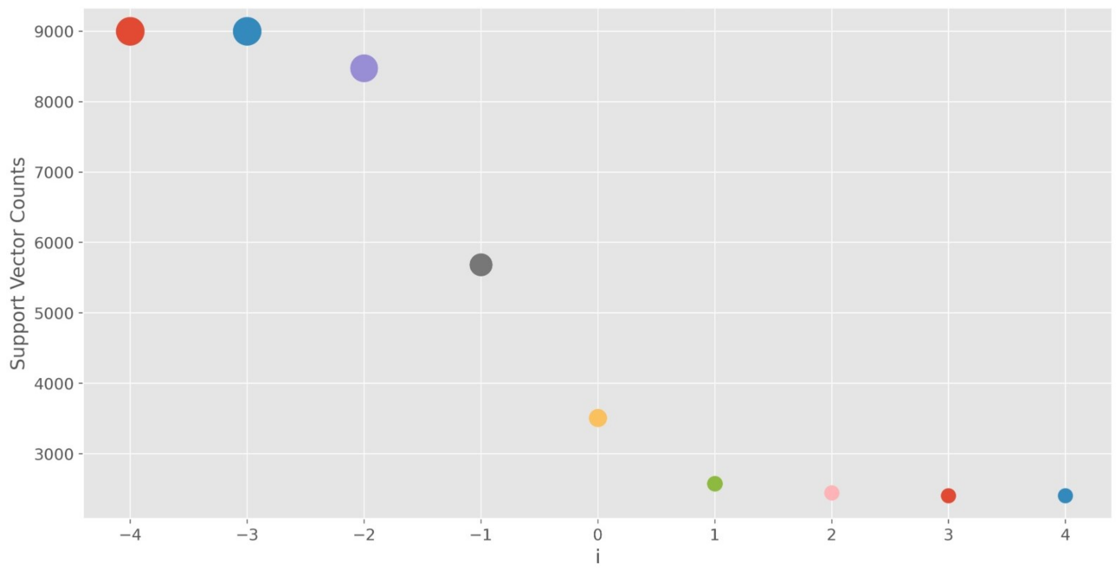


Figure 3: Number of support vectors for $c = 10^i$ for i in [-4,-3,..,3,4].

Part 2

Question 2a:
What is the best validation performance you are able to achieve with the quadratic kernel? what c value is used?

Response:
The best validation performance we were able to achieve (using accuracy as our performance met-

ric) was 91.72%. The value of C used was 0. This value was found by examining performance of test and validation data for c values in range $10^{[-4,-3,...,3,4]}$. After examining results showcased in Figure 4, we decided to explore the neighborhood of c in range [0,1]. Figure 4 highlights the results of that exploration where we isolated best c value for performance on validation accuracy.

Question 2b:
What trend do you theoretically expect to see for training and validation performance as we increase c? Plot the training and validation accuracy against different values of c. Does the trend you observe match your expectation?

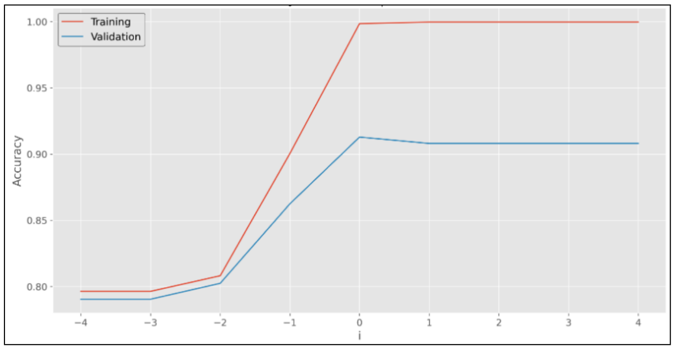


Figure 4: Accuracy of training and validation data for different c values used ($10^i \quad i \in [-4, -3, ..., 3, 4]$). for quadratic SVM

Response:
We theoretically thought the same thing would happen as in part 1 for training and validation accuracy as we increased c.

After plotting we did find similar results. We saw both training and validation accuracy increase as we increased the value of C. Then we saw validation performance hit a peak and then begin to decrease before leveling off. This happens for the same logic explained in part1b

Question 2c:
What relationship do we theoretically expect between c and the number of support vectors? Plot the number of support vectors against the different values of c. Does the trend you see match your theoretical expectations?

Response:
We expected the relationship between C and the number of support vectors to be the same pattern as explained in part 1c. We did observe our results following this trend (Figure 6). It follows this pattern for the same logic provided in part 1c.

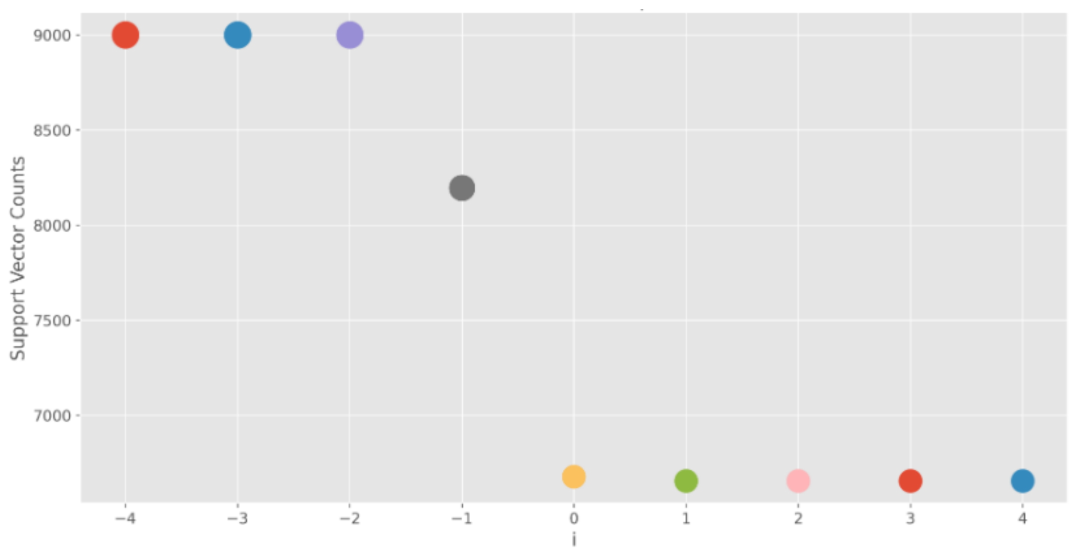


Figure 6: Number of support vectors for $c = 10^i$ for i in [-4,-3,..,3,4] of quadratic model.

Part 3

Question 3a:
What is the best validation performance you are able to achieve for RBF kernel? What c and γ parameters are used?

Response:
The best validation performance we were able to achieve was 92.12% with c = 10 and $\gamma = 0.02$. Exploration for γ in range $10^{[-5,-4,...,1]}$ and c values in range $10^{[-4,-3,...,3,4]}$ are showcased in heatmaps (Figure 7&8). Further exploration of refined c and γ values are showcased in heatmaps (Figure 9&10) in Appendix.

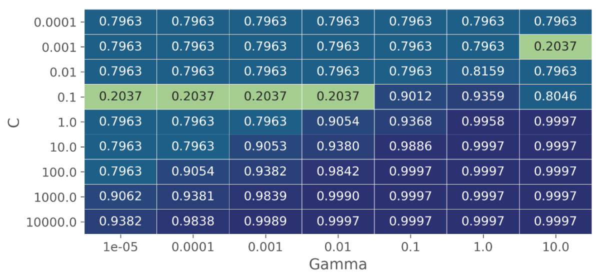


Figure 7: Training accuracy for γ in range $10^{[-5,-4,...,1]}$ and c in range $10^{[-4,-3,...,3,4]}$

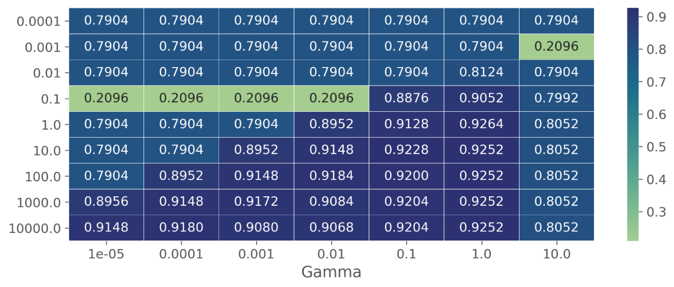


Figure 8: Validation accuracy for γ in range$10^{[-5,-4,...,1]}$ and c in range $10^{[-4,-3,...,3,4]}$

Question 3b:
What trend do you theoretically expect to see for training and validation performance as we in-

crease c with fixed γ? Does the trend you observe match your expectation?

Response:
We expect to see a similar pattern for the performance as seen in part1b and part2b. If we hold γ constant we will see both training and validation accuracy increase until hitting a point where training accuracy will increase while validation data decreases until both then remain constant. This trend is observed in our data. It can be seen in Figure 7 & 8 by comparing cell values in same γ value columns.

Question 3c:
What trend do you theoretically expect to see for training and validation performance as we decrease γ with fixed c? does the trend you observe match your expectation?

Response:
If we keep C fixed and increase γ we expect to see accuracy continue to increase for training data as γ values increase. We expect to see an increase of validation accuracy as γ increases until a certain value where validation accuracy will then begin to decrease. However this trend for validation only seems to be most pronounced at certain c values like 1

This general pattern is happening because γ essentially controls how much influence data points give to the decision boundary. So as you increase γ the decision boundary will take on more of the distinct shape of the data. So at high values of γ, variance will be high which will correlate to a decrease of validation accuracy (we begin over-fitting).

Question 3d:
With fixed γ What relationship do we theoretically expect between c and the number of support vectors? Plot the number of support vectors as a function of c value for $\gamma = 0.1$. does the trend you see match your theoretical expectations?

Response:
We would expect to see less support vectors as we increase the value of c. This is for the same logic explained in question 1c and 2c. As we increase c our margin will shrink and maybe shift some which in turn will decrease the amount of support vectors. This is the trend we observed shown in Figure 11.

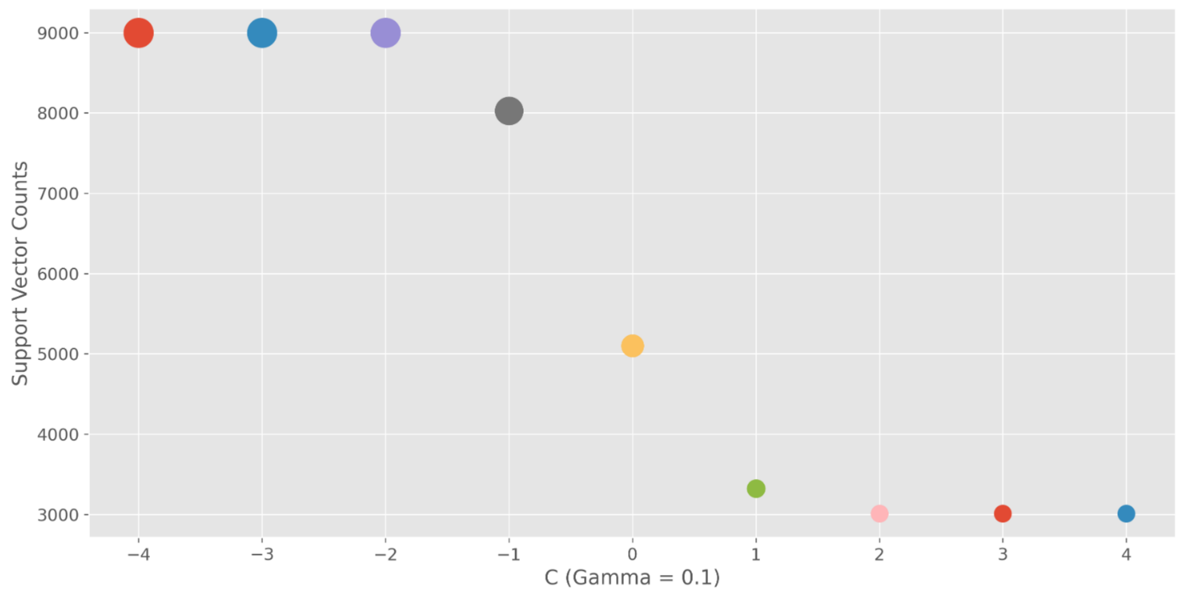


Figure 11: Number of support vectors for $c = 10^i$ for i in [-4,-3,..,3,4] of RBF model for fixed γ=0.1.

Question 3e:
With fixed c, what relationship do we theoretically expect between and the number of support vectors? Plot the number of support vectors as a function of γ for c = 10. Does the trend you see match your theoretical expectations?

Response:
We had a more difficult time determining what the trend would be for support vector count when fixing c and changing γ. As you increase γ the decision boundary will begin to take on the distinct shape of the data. So we figured the count of support vectors may not follow a clear trend, as the boundary shifts and shapes differently it might hit different clusters of data which could drastically change the count. We figured that the count of support vectors with changing γ would most likely be unique for each set of data.

Figure 12 showcases our finding. It seems to follow our logic somewhat – that there may be variability of SV count as γ changed the decision boundary. However, it seems to follow a quadratic trend, we thought the distribution may have looked more random. We would need to do some more testing to determine if it was a chance event that the distribution looked quadratic or whether that is usual for changing γ for fixed C of RBF kernel.

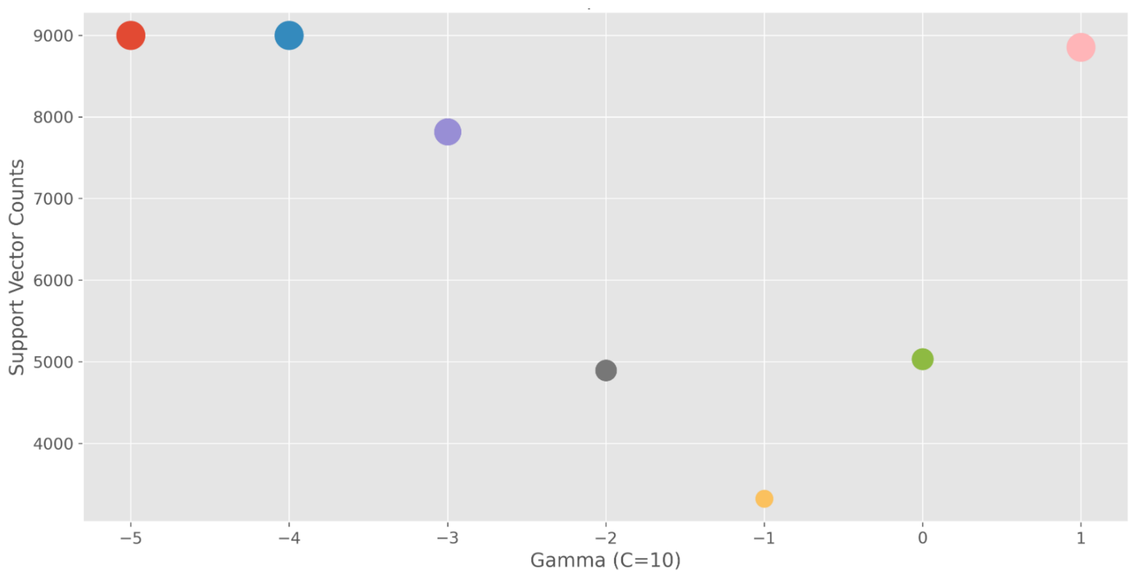


Figure 12: Number of support vectors for $c = 10$ with γ in [-5,-4,..,0,1] of RBF model.

Part 4 – Discussion

This discussion will include answers to question posed in part 4 of the PDF assignment and our own thoughts and observations.

Thinking about the four types of errors we decided we could rule out Bayes and optimization error as topics of much interest for this question. Bayes error will affect all the models and we have no control over it – it creates some type of lower bound of error for all models, and there is not much we can really do about it. We decided optimization error was not of much interest either. Since we were using the SKlearn functions we felt confident that these are probably solidly optimized.

The types of error that we decided vary more between the models were estimation and modeling error. Thinking about modeling error in general, as we increase complexity of our hypothesis class we will usually see a decrease in modeling error. So with that logic we can assume our modeling error decreases as we try linear, quadratic, and RBF respectively since we are increasing the complexity for each model.

Estimation error in general has to do with the fact that we are training on limited amounts of data. We figured this type of error would play a role in all three of our models – especially due to the nature of our data. We figured that not all tweets are going to have exactly the same words and would build a sparse but large spanning vocabulary feature bag. This would mean all our models are susceptible to some amount estimation error. One important thing to note is that we need to be careful of over fitting which would lead to higher estimation error. From this perspective, the more complex we make our models, like with quadratic and RBF, the more susceptible we are to estimation error.

One other observation we had as a group was for the RBF kernel it seemed like γ and C were almost proportional. When we decreased both c and γ by 10^-1 they maintained about the same training/validation accuracy.

Another important note is that we decided to balance our data. We decided to do this because we had so many more negative class samples. Before balancing our data we could, for example, predict negative class for every example and still get a high $\approx 89\%$ validation accuracy measure. After balancing our model we felt much better about the validity of our performance curves. We further expand on why and how we balanced our data below.

We found there was probably imbalance of the training data. The negative comments are present with approximately 0.71 ratio in the training set and 0.72 in the validation set. Initially, we were not aware of the issue until we observed some odd trends of the number of support vectors when C increases (Figure 13, Appendix). Further testing on other metrics confirmed that it was indeed an accuracy paradox. In particular, its validation recall is 0.47 while its balanced validation accuracy score (the average of recall obtained on each class) is only 0.73, which is slightly higher than blindly classifying all as negative (that would yield 0.72 validation accuracy).

Accuracy paradox is a situation in which a model has excellent accuracy that only reflects the underlying class distribution. It is often caused by imbalance data set, which is very common and expected. For example, the data set for fraudulent transaction classification is probably unbalanced because most of the transactions are not fraudulent.

One of the remedies for this problem is using cost-sensitive training, or class-weighted SVMs, in which costs for each class are different based on the ratio of them. Unsurprisingly, Scikit-learn does support this method to tackle this common problem. Training the model using SVC.fit() with the option class_weight="balanced" resulted in much better performance. Specifically, the model balanced validation accuracy went up to 0.82, whereas its recall increased to 0.65. Its validation accuracy was also improved from 0.89 to 0.92..

In general, there were not a lot of results that came as a shock to us. Once we realized about balancing our data, our performance curves and SV counts graphs all much better followed patterns that we thought we would expect to see. The only outcome we are still thinking about is the shape of the support vector count graph for fixed c varying γ in part 3. We will need to further discuss and think about why we the distribution looks the way it does in Figure 12.

Appendix

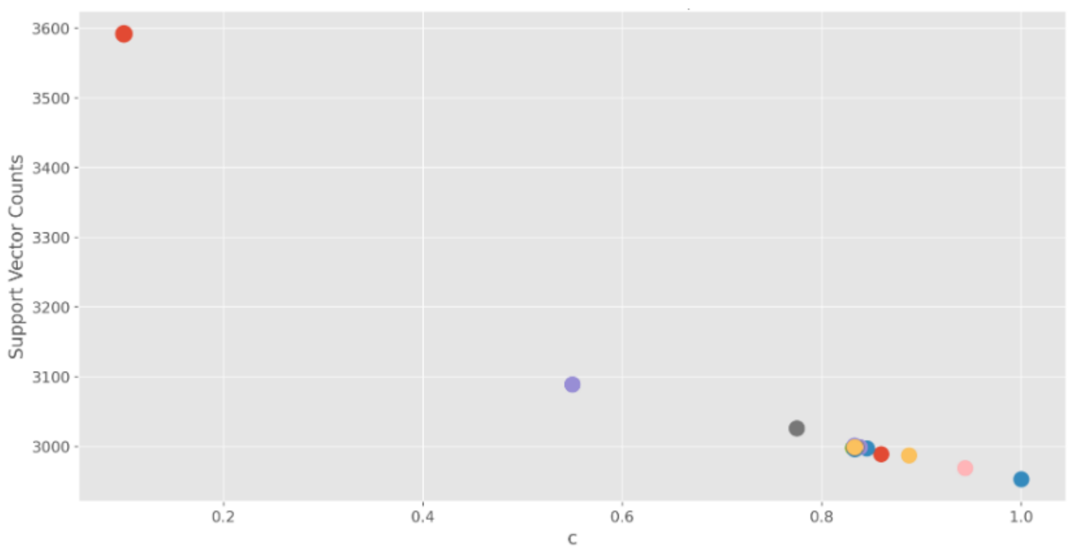
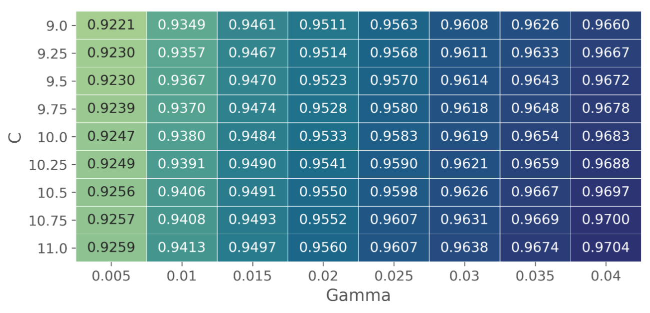
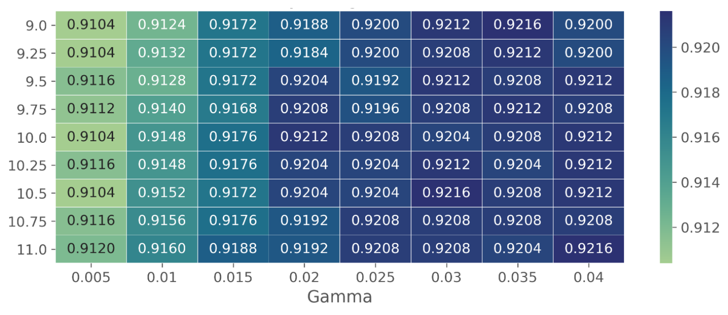


Figure 5: Trend in Number of support vectors for $c = 10^i$ for i in [0,1] for linear SVM



C \ Gamma	0.005	0.01	0.015	0.02	0.025	0.03	0.035	0.04
9.0	0.9221	0.9349	0.9461	0.9511	0.9563	0.9608	0.9626	0.9660
9.25	0.9230	0.9357	0.9467	0.9514	0.9568	0.9611	0.9633	0.9667
9.5	0.9230	0.9367	0.9470	0.9523	0.9570	0.9614	0.9643	0.9672
9.75	0.9239	0.9370	0.9474	0.9528	0.9580	0.9618	0.9648	0.9678
10.0	0.9247	0.9380	0.9484	0.9533	0.9583	0.9619	0.9654	0.9683
10.25	0.9249	0.9391	0.9490	0.9541	0.9590	0.9621	0.9659	0.9688
10.5	0.9256	0.9406	0.9491	0.9550	0.9598	0.9626	0.9667	0.9697
10.75	0.9257	0.9408	0.9493	0.9552	0.9607	0.9631	0.9669	0.9700
11.0	0.9259	0.9413	0.9497	0.9560	0.9607	0.9638	0.9674	0.9704

Figure 9: Refined training accuracy for γ in range [0.005,0.04] and c in range [9,11]



C \ Gamma	0.005	0.01	0.015	0.02	0.025	0.03	0.035	0.04
9.0	0.9104	0.9124	0.9172	0.9188	0.9200	0.9212	0.9216	0.9200
9.25	0.9104	0.9132	0.9172	0.9184	0.9200	0.9208	0.9212	0.9200
9.5	0.9116	0.9128	0.9172	0.9204	0.9192	0.9212	0.9208	0.9212
9.75	0.9112	0.9140	0.9168	0.9208	0.9196	0.9208	0.9212	0.9208
10.0	0.9104	0.9148	0.9176	0.9212	0.9208	0.9204	0.9208	0.9212
10.25	0.9116	0.9148	0.9176	0.9204	0.9204	0.9212	0.9204	0.9212
10.5	0.9104	0.9152	0.9172	0.9204	0.9204	0.9216	0.9208	0.9212
10.75	0.9116	0.9156	0.9176	0.9192	0.9208	0.9208	0.9208	0.9208
11.0	0.9120	0.9160	0.9188	0.9192	0.9208	0.9208	0.9204	0.9216

Figure 10: Refined training accuracy for γ in range [0.005,0.04] and c in range[9,11]

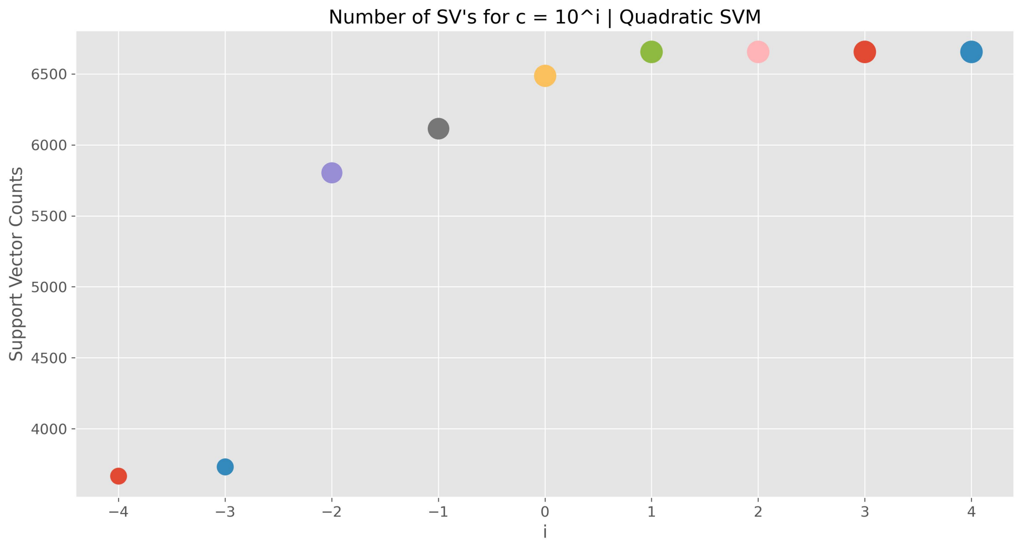


Figure 13: Trend in Number of support vectors for unbalanced data in quadratic model